# A Digital Hardware-based Approach for Molecular Sequence Comparison

Parag K. Lala

Department of Electrical Enginering, Texas A&M University – Texarkana, Texarkana, Texas 75503, USA
parag.lala@tamut.edu

***Abstract***

Several matching procedures have been proposed over the years to improve the search time of a query sequence in a database of molecular sequences. However due to the architectural limitations of conventional computers, only one sequence can be accessed/retrieved at a time from a database. The use of a parallel processing at least in theory, can provide a tremendous boost to the comparison of sequences. However, no algorithm employing parallel operations for this purpose is currently available. This paper presents a content addressable memory-based approach for fast comparison of molecular sequences.

***Keywords***

*Alignment; Parallel Processing; Content-Addressable Memory; Dynamic Algorithm; K-out-of-n Detector*

## Introduction

The functional information coded in genomic sequences has a variety of applications in molecular biology. Since similar sequences behave in a similar manner, the characteristics of a new sequence can be predicted by comparing it with known sequences. Sequence comparison provides an indication of which parts of comparing sequences are similar and which parts are different. It is employed to identify sequences similar to a given sequence from a database [Alt, Lip, Pea], to study phylogenetic relationships [Fen, Joh] etc. Sequence similarity in many ways is synonymous with the concept of sequence alignment, which identifies similarities and differences between sequences.

An alignment is a correspondence between the sequences, in which each symbol in a sequence is assigned to at most one of the symbols in the other sequence while maintaining the order of the symbols in the sequences. The main objective of the alignment is to have matching symbols at maximum number of positions. The alignment problem can be specified in terms of Hamming distance [Ham]. The Hamming distance between two binary sequences is defined to be the number of bits in which the sequences differ. For example the Hamming distance between the following sequences is 3:

```
1 0 1 0 1 0
|     | |
0 0 1 1 0 0
```

The concept of Hamming distance can also be extended to molecular sequences. For DNA sequences with alphabet (A,C,G,T), the Hamming distance could be used to identify the dissimilarities between them. For example, the Hamming distance of 4 between the following 6-digit sequences is 4:

```
T G A C G C
  | | | |
T A C G C C
```

When the distance is high the dissimilarity is high. Alternatively, lower distance value implies higher similarity. Thus the objective of the DNA sequence alignment is to derive a way to align sequences such that the distance between them is minimized. An alignment that exhibits the least dissimilarity is known as an *optimal alignment*. For example, the above 6-digit sequences can be aligned with Hamming distance of 2 as shown below; and this alignment is optimal:

```
T G A C G - C
  |       |
T - A C G C C
```

Differences in DNA sequences occur because of a series of evolutionary events, or due to errors known as *mutations*. Mutations are often due to radiation or other environmental conditions. Three types of mutations may occur in DNA sequences: *substitution*, *insertion* and *deletion*. Substitutions occur when one or more bases in a sequence are replaced by others. Kruskal [Kru] introduced the term *indel* to denote an insertion or a deletion of a base in a sequence.

It is often necessary to compare sequences of different lengths. This requires padding out the relevant sequences with null characters " —" (dash) till they have equal lengths. Apart from making the lengths equal, the dashes can be chosen such that regions of

subsequences from the comparing sequences are located at the same positions. This makes it easier to observe the similarities and differences between the sequences.

Sequences can be compared either by *global* or *local alignment*. Global alignment spans the whole length of comparing sequences and is appropriate if the sequences are likely to share substantial similarity. The alignment attempts to match them to each other from end to end, even though parts of the sequences may not match. Local alignment is used to identify common sub-regions of similarity between long sequences. There is no attempt to force entire sequences into an alignment, just those parts that appear to have good similarity.

Thus local alignment can be used to identify similarities between sequences without having a priori knowledge of the presence of any, and is therefore particularly useful for comparison of long DNA sequences where only small subsequences may be related. Although it may seem that local alignments should always be preferred to global alignments, it is difficult to spot an overall similarity if only local alignments are employed; global alignments are useful in some cases.

An alignment between two sequences *A* and *B* over an alphabet A can be represented by a matrix of two rows and *c* columns, where *c* is the length of the longer of the two sequences. The first (upper) row is the source sequence *A*, and the second (lower) row is the target sequence *B*. Two identical letters at the same columns in a pair of rows form a *match*, and two different ones form a *mismatch*. A " — " in sequence (row) *A* aligned with a letter in sequence (row) *B* can be viewed as an insertion of the letter in sequence *B*, or as a deletion of the letter from sequence *A*.

Let us illustrate the concept of alignment by comparing sequences *A* = TGACG and *B* = TGGCG Two possible alignments of the sequences are as follows:

***A*** = T G A C – G ***A*** = T G A C –G
***B*** = T – G CA G ***B*** = T G – C A G

The second alignment is preferable to the first because the first one has three matches, one substitution and three i*ndels*, whereas the second one has four matches and two *indels*. The placement of *indels* to minimize the distance between two sequences being aligned is a complicated problem.

## Dynamic Algorithm Based Matching

The number of possible alignments for two sequences of length *m* and *n* is extremely large. Therefore the obvious solution of enumerating all alignments and then choosing the one with the smallest or the highest score (depending on the scoring scheme used) is computationally impractical. An efficient alignment process needs to employ a completely automatic method e.g. dynamic programming algorithms, which is usually used to solve optimization problems. A standard dynamic algorithm for sequence matching consists of two steps, in the first of which the score for the optimal alignment is computed while in the other the alignment corresponding to this score is derived.

The data structure used by dynamic programming algorithms for sequence alignment is the *distance matrix*. For two sequences A and B of length *m* and *n*, a distance matrix D consists of *m* rows and *n* columns. Each entry $d_{ij}$ in the matrix is computed using a scoring scheme. There are many different possible scoring schemes. Once a distance matrix has been constructed a dynamic programming method can be employed to compute the optimal score of an alignment between the two sequences. Needleman and Wunsch [Nee] are the first to propose such a method. Their motivation for developing the method is to maximize similarities between amino acid sequences allowing global comparison of an entire query sequence with all sequences in a database. One drawback of this global alignment algorithm is that highly similar shorter subsequences with meaningful similarities may be ignored because of the overall objective of matching the largest number of residues of one sequence with another sequence.

Smith and Waterman [Smi] proposed an algorithm, perhaps the most widely used local similarity algorithm for biological sequence comparison, based on the concept of dynamic programming. It identifies pairs of subsequences of all possible lengths in the query and the database sequence that have maximum degree of similarity. As in the Needleman et al.'s algorithm the alignment of the sequences is done by the substitution of one character by another, and by proper placement of indels. Positive score is given for exact matches, and negative scores to mismatches. The minimum score recorded in the matrix must be zero. Unlike the Needleman-Wunsch algorithm, an optimal path in the Smith-Waterman algorithm can begin and end anywhere in the matrix. Thus the entire matrix has to be searched for subsequences of high

similarity,indicated by a high score.

Two other dynamic programming algorithms BLAST [Alt] and FASTA[Lip] have been developed to identify possible homologues for a query sequence in a database of all other known sequences. Both BLAST and FASTA employ heuristic techniques.Although it has not been proved that BLAST is faster than FASTA, BLAST based software packages are extensively used to search sequence databases for local alignments [Sot].

## Hardware-Based Comparison

In recent years there have been some academic and industrial efforts on the use of FPGAs (Field Programmable Gate Arrays) to enhance the speed of sequence matching [Sot, Oli, Bro, Caf, Sot, Alj, Llo]. However, all the techniques developed so far for accelerating this task sequentially compare a query sequence with sequences stored in a conventional memory system. Although this strategy results in improvement in the matching speed compared to traditional software-based algorithms, the comparison of the query sequence with the stored sequences still needs to be done one sequence at a time because of the sequential nature of information retrieval from the memory system. Significant improvement in the matching speed can be achieved if a query sequence can be compared with all the stored sequences in parallel, and if all the matched sequences can be accessed simultaneously. This paper presents a digital hardware-based sequence matching procedure based on this principle; an earlier and condensed version of the paper has been published in [Lal].

It is assumed that a query sequence is a subsequence of one or more sequences stored in computer memory system. The bases and the *gap* in a sequence are represented by binary patterns as shown below:

A = 000, C = 010, G = 100, T = 110, – (*dash*) = $xx1$

where $x$ is a *don't care*. Thus, a 1 in the least significant bit of a 3-bit binary representation indicates a *dash*.

Fig.1 shows the block diagram of the proposed matching system. It consists of a dedicated Content Addressable Memory (CAM) block and a bi-directional barrel shift register. A CAM unlike a traditional RAM (Random Access Memory) is addressed by the desired content, and an address that stores the content is obtained as the output of the CAM [Jal]. It is assumed that a CAM-based memory system will store a large number of sequences, and the content of a stored sequence has to have a few common codons i.e. all three bases identical, for it to be accessible. However, in order to avoid a large number of *hits* in the CAM, the number of codons considered to match has to be properly selected to keep the number of accessed sequences to a realistic number. For the sake of simplifying the explanation of our approach we consider matching of only one codon to access relevant sequences in a CAM.

While comparing a query sequence with the stored sequences in a CAM, a better match may be obtained by shifting the query sequence left (or right) by one or more bases at a time, and then comparing it with the stored sequences. Traditionally, the alignment process starts after some partially-matched sequences have been retrieved from the memory block. Then by proper placement of *dashes* in the query sequence, matching of bases in the comparing pair is maximized. In the proposed hardware-based sequence matching approach, one of the goals is to retrieve sequences with a high degree of similarity whenever possible. The possibility of better matching is explored by shifting the query sequence to the left (or right) before the stored locations are accessed. The shifting of a single base in the query sequence requires simultaneous shifting of three bits.

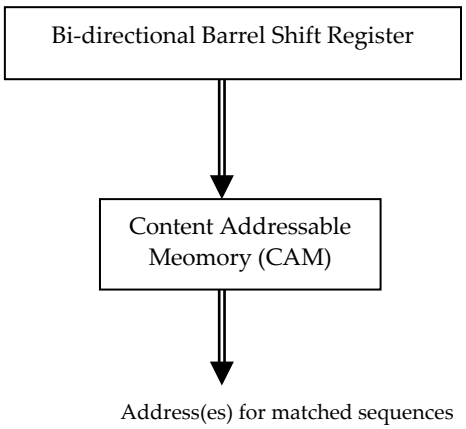


FIG.1 HARDWARE-BASED SYSTEM FOR SEQUENCE MATCHING

The barrel shifter in Fig.1 is used to shift pre-selected multiple data bits and to store the query sequence with length $3b$ where $b$ is the number of bits in the sequence. It has a shift left and shift right capability, and can be used to shift three bits i.e. one base at a time. It is assumed that the shift-in data is 001 i.e. a *dash*. Although the proposed system is designed for local sequence alignment, it could also be used for global sequence alignment.

The proposed architecture allows simultaneous comparison of a pre-defined number of codons in a

query sequence with the same number of codons in stored sequences in the CAM. Fig. 2 shows the CAM-architecture. If a majority of the codons in the query sequence match with the corresponding codons in the stored sequences then the addresses of these stored sequences are retrieved; and they identify the locations of sequences that partially match the query sequence. The address corresponding to a matched location is activated via the output of an *k-out-of-n* detector which produces an output of 1 if at least *k* out of *n* inputs are at 1. Thus a 2-out-of-3 detector will produce an output of 1 if two or more of its inputs are at 1. A 2-out-of-3 detector can also be considered as a *majority voter*.

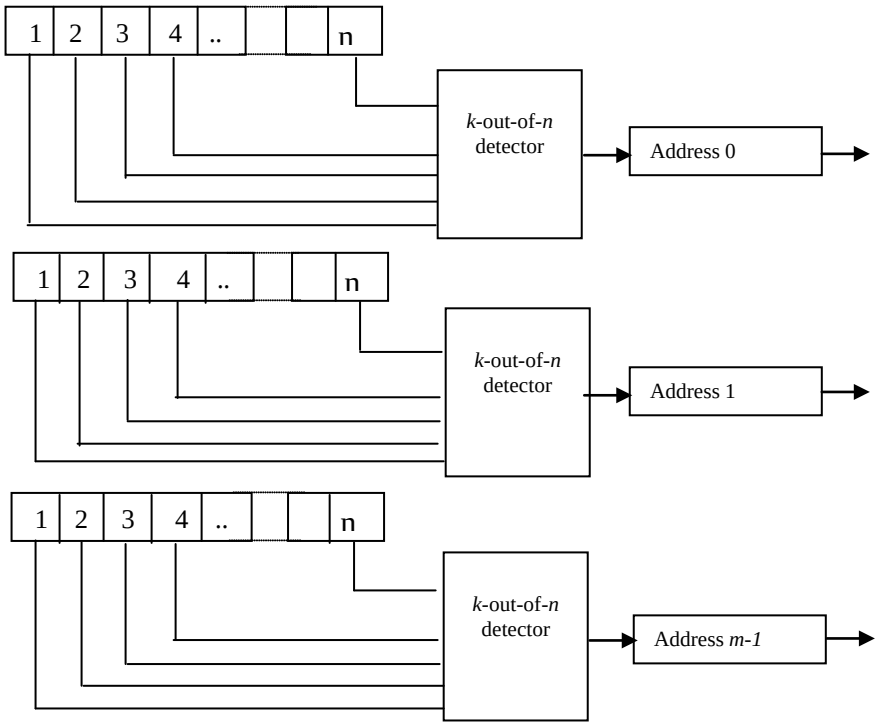


FIG. 2 PROPOSED *M* X *N* CAM ARCHITECTURE

A *k*-out-of-*n* detector in Fig. 2 is used to enable the address corresponding to a matched location in the CAM. For example, if the first five codons of a query sequence are compared with a stored sequence and a match is assumed if any three of the five codons are similar, then a 3-out-of-5 detector is used to identify the address of the stored location. To illustrate it is assumed that the first five codons in the following six codon query sequence

ACG AT- CGT –GA TCG ATG

are compared with a stored sequence

ACG CAG CGT TTC TCG AC- C-T ATC

Since three codons in the comparing sequences match, a 3-out-of-5 detector circuit will produce an output of 1. On the other hand, if four codons have to be similar for a match , then a 4-out-of-5 detector has to be used; this detector will produce an output of 0 in this case. Thus, a programmable detector that allows pre-selection of the *k* value in the *k*-out-of-*n* detectors will enable comparison of pre-selected parts of the query sequence to stored sequences.

The CAM architecture as shown in Fig. 2, has *m* rows (addresses) and *n* columns (codons). Each row consists of *n* comparators, *n* two-input AND gates, and a *k*-out-of-*n* programmable detector. One of the inputs to a two-input AND gate is the output of a comparator while the other input is programmable. The programmable input to the AND gate is set to be 1 if a codon in the query sequence is matched with the codon at the identical postion in the stored sequence.

A matching circuit is used to compare a codon in the query sequence with that in a stored sequence. Fig. 3. shows the circuit for matching of two codons where *lmn* and *pqr* are the codons in the query sequence and a stored sequence respectively. Note that this circuit includes a 2-out-of-3 detector. Thus as long as two bases in the comparing codons match, the codons are assumed to be matched. The output of the 2-out-of-3 detector is connected to the input of the programmable detector associated with the stored sequence address decoder via a two-input AND gate.

The implementation of the comparator in the matching circuit of Fig. 3 is shown in Fig.4. The comparator is composed of 9 EX-NOR gates and three AND gates. The outputs of the AND gates feed the 2-out-of-3 detector. To illustrate the application of the system of Fig.1 for sequence comparison let us assume the following query sequence in the barrel shifter:

ACT –GAT-CGAA

Suppose that the contents of the CAM (assuming it has four locations) are as below:

| | Address |
|---|---|
| A-CT-GTCGCGA | 00 |
| ACTG-CT-GGAA | 01 |
| -CTATGT-CACT | 10 |
| -AC—G—T--AG | 11 |

If *k* is selected to be 3 for the detectors in Fig. 2 i.e. 3-out-of-4 (*n*=4 in this example) then only address 01 will be identified as the CAM location that contains the sequence with most codons. However if *k*=2 (i.e. 2-out-of-4 detector) is used instead, both addresses 01 and 10 will be identified.

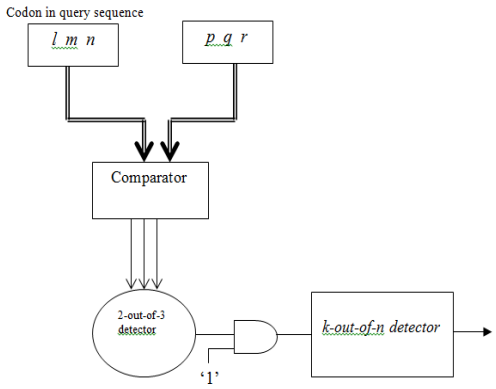


FIG. 3. MATCHING CIRCUIT

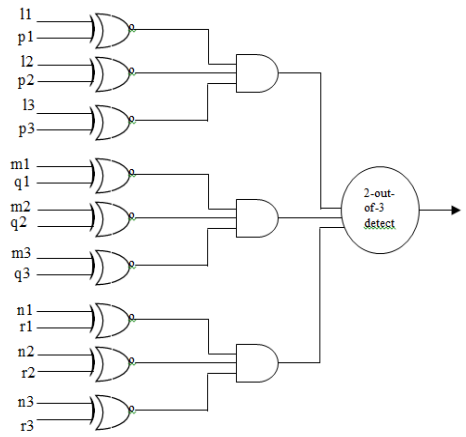


FIG.4 COMPARATOR CIRCUIT

One particular advantage of the proposed strategy is that it simplifies the identification of shorter subsequences that may be common among several stored sequences. To illustrate, it is determined whether – – TAAG is part of the following stored sequences:

| Address | Contents |
|---|---|
| 00 | ATCT –A –CAGCG |
| 01 | –TAAGC –CA GAG |
| 10 | –TGCTAAGC TGAT |

First – – TAAG is transferred to the barrel shift register as the query sequence. As shown below, the subsequence occupies the first six most significant positions in the barrel shifter, and the remaining positions are filled with *dashes*.

– – TAAG – – – – – –

Since no matching subsequences are detected in any of the stored sequences, the barrel shifter is shifted to the left by one base:

– TAAG – – – – – – – –

This results in matching with the contents of address 01.

A shift to the right by two bases

– – – – TAAG – – – –

will result in matching with the contents of address 10.

In certain cases a desired subsequence may appear in a stored sequence in bits and pieces. For example, it is not immediately obvious that the above subsequence is part of the sequence at location 00. However, a number of shift operations of the query sequence as shown below verify its presence:

– – – TAAG – – – – –

– – – TA –AG – – – –

– – – T–A–AG – – – –

– – – T–A– –AG – –

Certain shift operations in this case require shifting of individual bases in the query sequence. Similar situation will arise when a query sequence has certain similarity with a stored sequence. Once the stored sequence has been retrieved, a number of shift operations of the individual bases or subsequences in the query sequence may be needed to increase the number of positions with matching symbols. Thus a major objective will be how to design the barrel shifter

such that one or more bases can be shifted left or right without necessarily shifting the whole query sequence.

## Conclusions

Currently available tools for molecular sequence matching are in general software-based. The computation time needed to match is dependent on search sensitivity. A low sensitivity search requires short computation time, whereas for a high sensitivity search the computation time is very long. The conventional i.e. Von Neumann architecture based computers can process information only serially, thus limiting their speed of computation. This paper presented a digital hardware-based sequence matching technique that allows simultaneous comparison of a query sequence with all the stored sequences in a database, thereby achieving significant improvement in the matching speed compared to software-based techniques. Although the proposed technique focuses on DNA sequences, it may also be possible to extend the technique to protein databases.

## ACKNOWLEDGEMENT

The work presented in this paper was in part supported by the National Science Foundation under Grant No. 0925080.

## REFERENCES

Altschul,S.F ., W.Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic Local alignment Search Tool", Jour. Molecular Biology, vol.215(3), pp.403- 410, 1990

Al Junid, S. A., M. A. Haron, Z. Abd Majid, F. N. Osman, H. Hashim M. F. Idros and M. R. Dohad, "Optimization of DNA Sequences Data to Accelerate DNA Sequence Alignment on FPGA", Proc. Fourth Asia International Conf. on Mathematical/Analytical Modelling and Computer Simulation, Malyasia, pp.231- 236, 2010

Brown, B.O., M. Yin, and Y. Cheng," DNA Sequence Matching Processor Using FPGA and JAVA Interface", Proc. 26th Annual International Conference of the IEEE EMBS, pp.3043-3046, 2004

Cafferena, G, C.Pederira, C.Carreras, S.Bojanic and O.Nieto-Taladriz,"FPGA acceleration for DNA sequence alignment", Jour. of Circuits, Systems, and Computers (JCSC), Vol.16, Issue 2, pp. 245- 266, 2007

Feng, D.F., and R.F. Dolittle, " Progressive sequence alignment as a prerequisite to correct phylogenetic trees", Jour. Mol. Evol., vol.25, pp.351-360, 1987.

Hamming, R.W.,, "Error detecting and error correcting codes", Bell Syst. Tech. Jour., vol. 24(2), pp. 147-160, 1950.

Jalaeledine, S.M., and L.G. Johnson, "Associative IC memories with relational search and nearest match capabilities", IEEE Jour. Solid. State Circuits, vol.27, no.6, pp. 892- 900,1992.

Johnson, S., M.J.Sutcliffe and T.L.Blundell,"Molecular anatomy: Phyletic relationships derived from three-dimensional structures of proteins", Jour. Mol. Evol., vol.30, pp.43-59, 1990.

Kruskal, J.B., "An overview of sequence comparison", Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, pp.1-44, Addison Wesley, 1983.

Lala, Parag K and J. Parkerson,"A CAM(Content Addressable Memory)-based architecture for molecular sequence matching", Intl. Conf. Bioinformatics and Computational Biology, Las Vegas, July 18-21, 2011

Lipman, D. J. Lipman and W.R.Pearson, "Rapid and protein similarity searches". Science 227 (4693), pp. 1435–41, 1985

Lloyd, S., and Q.O. Snell, "Hardware Accelerated Sequence Alignment with Traceback", International Journal of Reconfigurable Computing,Vol. 2009, Article ID 762362,, 10 pages, 2009

Needleman, S.D., and C.D. Wunsch., " A general method applicable to the search for similarities in the amino acid sequences of two proteins", Jour. Mol. Biol., vol.48, pp .443-453, 1970

Oliver,T., B. Schmidt, D. Maskell, D. Nathan, and R. Clemens, "High-speed Multiple Sequence Alignment on a reconfigurable platform", International Journal of Bioinformatics Research and Applications, Vol. 2, No.4, pp. 394 – 406, 2006

Pearson ,W.R., "Using the FASTA program to search protein and DNA sequence data base", Methods in Molecular Biology, 25, pp.365--389, 1994

Smith.,T.F., and M.S.Waterman, "Identification of common molecular subsequences", Adv. Appl. Math., vol.2, pp.482- 489, 1981.

Sotiriades, E., C. Kozanitis and A. Dollas, "FPGA based Architecture for DNA Sequence Comparison and

Database Search", Proc.20th IEEE International Parallel & Distributed Processing Symposium, pp 186-, 2006

Sotiriades,E., and A. Dollas, "A General Reconfigurable Architecture for the BLAST Algorithm", Journal of VLSI Signal Processing_Systems, vol.48, issue 6, pp.189-208, 2007

**Parag K. Lala** received an M.Sc.(Eng.) degree from King's College, University of London and a Ph.D. from The City University of London. His current research innterests inclue On-line testable logic synthesis, Quantum Computing & Cryptography and Biologically-inspired digital system design.

He is the Cary and Lois Patterson Professor of Electrical Engineering at Texas A&M University-Texarkana. Previous to his current position he was the Thomas Mullins Chair Professor of Computer Engineering, at University of Arkansas, Fayetteville. He is the author/co-author of more than 140 papers and 7 books.

Dr.Lala was selected Outstanding Educator in 1994 by the Central North Carolina section of the IEEE. In 1998 he was awarded a D.Sc.(Electrical Eng.) degree by the University of London for for his contributions to digital hardware design and testing, and self-checking logic design. He was made a Fellow of the IEEE in 2001 for contributions to the development of self checking logic and associated checker design. He is also a Fellow of the IET (previously IEE) in the UK.